AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Previously Presented) A method of executing a sequence of variable length instructions stored prior to any fetching as part of decoding and execution within a plurality of discrete memory address regions within a memory of a data processing apparatus, said plurality of discrete memory address regions including a current memory address region and a following memory address region, said current memory address region and said following memory address region being non-contiguous with a gap therebetween, said method comprising the steps of:

(i) detecting an attempt to execute a variable length instruction spanning two discrete memory address regions, said two discrete memory address regions being said current memory address region and said following memory address region and said attempt triggering a memory abort, said detecting together with said triggering causing further steps (ii)-(iv)to be performed;

(ii) copying instruction data from an end portion of said current memory address region and a start portion of said following memory address region into a fix-up memory address region of said memory to form concatenated instruction data containing said variable length instruction, said fix-up memory address region being separate from said current memory address region and said following memory address region within said memory;

(iii) diverting program execution flow to execute said current variable length instruction from within said concatenated instruction data in said fix-up memory address region; and

(iv) restoring program execution flow to execute instructions following said variable length instruction from within said following memory address region.

2. (Currently Amended) A method as claimed in claim 1, wherein said ~~steps~~ step of detecting is performed under hardware control.

3. (Original) A method as claimed in claim 1, wherein said steps of concatenating, diverting and restoring are performed under software control.

4. (Original) A method as claimed in claim 1, wherein variable length instructions are fetched from said memory to an instruction buffer before being executed.

5. (Original) A method as claimed in claim 4, wherein said step of detecting occurs as said variable length instruction is read from said instruction buffer.

6. (Original) A method as claimed in claim 4, wherein fetching of variable length instructions to said instruction buffer is performed by fetching instruction data from sequential memory addresses under hardware control.

7. (Original) A method as claimed in claim 6, wherein when a fetch is attempted from a memory address beyond an end point of said current memory address region to a buffer memory location, said buffer memory location is marked as not containing valid instruction data.

8. (Original) A method as claimed in claim 7, wherein said step of detecting comprises detecting an attempt to execute an instruction at least partially stored in a buffer memory location marked as not containing valid instruction data.

9. (Original) A method as claimed in claim 1, wherein a program counter value specifies a location of a variable length instruction to be executed.

10. (Original) A method as claimed in claim 9, wherein said steps of diverting and restoring act by modifying said program counter value.

11. (Original) A method as claimed in claim 1, comprising setting a single step flag, said single step flag serving to limit hardware execution of variable length instructions to a single variable length instruction.

12. (Original) A method as claimed in claim 11, wherein said single step flag serves to limit hardware execution of variable length instructions to a single variable length instruction from said concatenated instruction data to a single variable length instruction before control is returned to software to perform said step of restoring.

13. (Original) A method as claimed in claim 11, wherein upon hardware execution of said single variable length instruction, said single step flag is cleared under hardware control.

14. (Original) A method as claimed in claim 11, wherein said single step flag is stored within a coprocessor register.

15. (Original) A method as claimed in claim 1, comprising calculating a start address within said following region of memory of a following variable length instruction following said current variable length instruction.

16. (Original) A method as claimed in claim 15, wherein said step of calculating uses as inputs a start address of said following memory region and a program counter value pointing to said following variable length instruction within said fix-up memory region following execution of said current variable length instruction.

17. (Original) A method as claimed in claim 16, comprising storing said start address of said following memory region before said step of diverting.

18. (Original) A method as claimed in claim 1, wherein said variable length instructions are Javacard bytecode instructions executed as native instructions by said data processing apparatus.

19. (Original) A method as claimed in claim 18, wherein said data processing apparatus also supports execution of instructions of a further instruction set, said steps of concatenating, diverting and restoring being performed under control of instructions of said further instruction set.

20. (Original) A method as claimed in claim 19, wherein said steps of diverting and restoring are performed using state switching branch instructions that serve to switch to execution of Java bytecodes starting from a specified memory address location.

21. (Original) A method as claimed in claim 1, wherein a program to be executed is stored within fragmented memory regions within said memory.

22. (Currently Amended) A hardware apparatus for executing a sequence of variable length instructions stored prior to any fetching as part of decoding and execution within a plurality of discrete memory address regions within a memory, said plurality of discrete memory address regions including a current memory address region and a following memory address region, said current memory address region and said following memory address region being non-contiguous with a gap therebetween, said apparatus comprising:

(i) a detector configured to detect an attempt to execute a variable length instruction spanning two discrete memory address regions, said two discrete memory address regions being said current memory address region and said following memory address region and said attempt triggering a memory abort;

(ii) combining logic circuitry configured, when said detector detects said attempt and said memory abort, to copy instruction data from an end portion of said current memory address region and a start portion of said following memory address region into a fix-up memory address region of said memory to form concatenated instruction data containing said

variable length instruction, said fix-up memory address region being separate from said current memory address region and said following memory address region within said memory;

(iii) diverting logic circuitry configured to divert program execution flow to execute said current variable length instruction from within said concatenated instruction data in said fix-up memory address region; and

(iv) restoring logic circuitry configured to restore program execution flow to execute instructions following said variable length instruction from within said following memory address region.

23. (Original) Apparatus as claimed in claim 22, wherein said detector is non-programmable hardware.

24. (Previously Presented) Apparatus as claimed in claim 22, wherein said concatenating logic, said diverting logic circuitry and said restoring logic circuitry comprise programmable hardware operating under software control.

25. (Original) Apparatus as claimed in claim 22, wherein variable length instructions are fetched from said memory to an instruction buffer before being executed.

26. (Original) Apparatus as claimed in claim 25, wherein said detector acts as said variable length instruction is read from said instruction buffer.

27. (Original) Apparatus as claimed in claims 25, wherein fetching of variable length instructions to said instruction buffer is performed by fetching instruction data from sequential memory addresses under hardware control.

28. (Original) Apparatus as claimed in claim 27, wherein when a fetch is attempted from a memory address beyond an end point of said current memory address region to a buffer memory location, said buffer memory location is marked as not containing valid instruction data.

29. (Previously Presented) Apparatus as claimed in claim 28, wherein said detector is configured to detect an attempt to execute an instruction at least partially stored in a buffer memory location marked as not containing valid instruction data.

30. (Original) Apparatus as claimed in claim 22, wherein a program counter value specifies a location of a variable length instruction to be executed.

31. (Previously Presented) Apparatus as claimed in claim 30, wherein said diverting logic circuitry and said restoring logic circuitry are configured to act by modifying said program counter value.

32. (Original) Apparatus as claimed in claim 22, comprising means for setting a single step flag, said single step flag serving to limit hardware execution of variable length instructions to a single variable length instruction.

33. (Currently Amended) Apparatus as claimed in claim 32, wherein said single step flag serves to limit hardware execution of variable length instructions to a single variable length instruction from said concatenated instruction data to a single variable length instruction before control is returned to software to perform said ~~step of~~ restoring.

34. (Original) Apparatus as claimed in claim 32, wherein upon hardware execution of said single variable length instruction, said single step flag is cleared under hardware control.

35. (Original) Apparatus as claimed in claim 32, wherein said single step flag is stored within a coprocessor register.

36. (Previously Presented) Apparatus as claimed in claim 22, comprising calculating logic configured to calculate a start address within said following region of memory of a following variable length instruction following said current variable length instruction.

37. (Previously Presented) Apparatus as claimed in claim 36, wherein said calculating logic circuitry is configured to use as inputs a start address of said following memory region and a program counter value pointing to said following variable length instruction within said fix-up memory region following execution of said current variable length instruction.

38. (Previously Presented) Apparatus as claimed in claim 37, comprising means for storing said start address of said following memory region before said diversion.

39. (Currently Amended) Apparatus as claimed in claim 22, wherein said variable length instructions are Java bytecode instructions executed as native instructions by said ~~data processing~~ apparatus.

40. (Currently Amended) Apparatus as claimed in claim 39, wherein said ~~data processing~~ apparatus also supports execution of instructions of a further instruction set, and wherein said concatenating logic circuitry, said diverting logic circuitry, and said restoring logic circuitry are under control of instructions of said further instruction set.

41. (Previously Presented) Apparatus as claimed in claim 40, wherein said diverting logic circuitry and said restoring logic circuitry are configured to use mode switching branch instructions that serve to switch to execution of Java bytecodes starting from a specified memory address location.

42. (Original) Apparatus as claimed in claim 22, wherein a program to be executed is stored within fragmented memory regions within said memory.

43. (Previously Presented) A computer program product including a storage medium encoded with instruction code readable by a data processing apparatus, which when executed by the data processing apparatus, controls the data processing apparatus to execute a sequence of variable length instructions stored prior to any fetching as part of decoding and execution within

a plurality of discrete memory address regions within a memory of said data processing apparatus, said plurality of discrete memory address regions including a current memory address region and a following memory address region, said current memory address region and said following memory address region being non-contiguous with a gap therebetween, said instruction code comprising:

code configured after an attempt to execute a variable length instruction spanning two discrete memory address regions, said two discrete memory address regions being said current memory address region and said following memory address region and said attempt triggering a memory abort, said code including:

(i) concatenating code encoded in the storage medium which, when executed by the data processing apparatus in response to said attempt and said memory abort, controls the data processing apparatus to copy instruction data from an end portion of said current memory address region and a start portion of said following memory address region into a fix-up memory address region of said memory to form concatenated instruction data containing said variable length instruction, said fix-up memory address region being separate from said current memory address region and said following memory address region within said memory;

(ii) diverting code encoded in the storage medium which, when executed by the data processing apparatus, controls the data processing apparatus to divert program execution flow to execute said current variable length instruction from within said concatenated instruction data in said fix-up memory address region; and

(iii) restoring code encoded in the storage medium which, when executed by the data processing apparatus, controls the data processing apparatus to restore program execution

flow to execute instructions following said variable length instruction from within said following memory address region.

44. (Previously Presented) A computer program product as claimed in claim 43, wherein detection of said attempt to execute a variable length instruction spanning two discrete memory address regions is performable under hardware control.

45. (Previously Presented) A computer program product as claimed in claim 43, wherein when the code is executed by the data processing apparatus, the data processing apparatus fetches variable length instructions from said memory to an instruction buffer before being executed.

46. (Previously Presented) A computer program product as claimed in claim 45, wherein when the code is executed by the data processing apparatus, said detection occurs as said variable length instruction is read from said instruction buffer.

47. (Previously Presented) A computer program product as claimed in claim 45, wherein when the code is executed by the data processing apparatus, the data processing apparatus fetches variable length instructions to said instruction buffer by fetching instruction data from sequential memory addresses under hardware control.

48. (Previously Presented) A computer program product as claimed in claim 47, wherein when the code is executed by the data processing apparatus and a fetch is attempted from a memory address beyond an end point of said current memory address region to a buffer memory location, the data processing apparatus marks said buffer memory location as not containing valid instruction data.

49. (Original) A computer program product as claimed in claim 48, wherein said detection comprises detecting an attempt to execute an instruction at least partially stored in a buffer memory location marked as not containing valid instruction data.

50. (Previously Presented) A computer program product as claimed in claim 43, wherein when the code is executed by the data processing apparatus, a program counter value specifies a location of a variable length instruction to be executed.

51. (Previously Presented) A computer program product as claimed in claim 50, wherein when the code is executed by the data processing apparatus, said diverting code and said restoring code act by modifying said program counter value.

52. (Previously Presented) A computer program product as claimed in claim 43, comprising setting code configured, when executed by the data processing apparatus, to set a single step flag, said single step flag serving to limit hardware execution of variable length instructions to a single variable length instruction.

53. (Currently Amended) A computer program product as claimed in claim 52, wherein said single step flag serves to limit hardware execution of variable length instructions to a single variable length instruction from said concatenated instruction data to a single variable length instruction before control is returned to software to perform said ~~step of~~ restoring.

54. (Previously Presented) A computer program product as claimed in claim 52, wherein upon hardware execution of said single variable length instruction, and the code is executed by the data processing apparatus, said single step flag is cleared under hardware control.

55. (Previously Presented) A computer program product as claimed in claim 52, wherein when the code is executed by the data processing apparatus, said single step flag is stored within a coprocessor register.

56. (Previously Presented) A computer program product as claimed in claim 43, comprising calculating code encoded in the storage medium which, when executed by the data processing apparatus, controls the data processing apparatus to calculate a start address within said following region of memory of a following variable length instruction following said current variable length instruction.

57. (Previously Presented) A computer program product as claimed in claim 56, wherein said calculating code which, when executed by the data processing apparatus, controls the data processing apparatus to use as inputs a start address of said following memory region and a program counter value pointing to said following variable length instruction within said fix-up memory region following execution of said current variable length instruction.

58. (Previously Presented) A computer program product as claimed in claim 57, wherein when the code is executed by the data processing apparatus, said start address of said following memory region is stored before said diversion.

59. (Previously Presented) A computer program product as claimed in claim 43, wherein said variable length instructions are Java bytecode instructions executed as native instructions by said data processing apparatus.

60. (Previously Presented) A computer program product as claimed in claim 59, wherein said data processing apparatus also supports execution of instructions of a further instruction set, and when the code is executed by the data processing apparatus, said

concatenating, diverting and restoring are performed under control of instructions of said further instruction set.

61. (Previously Presented) A computer program product as claimed in claim 60, wherein said diverting code and said restoring code are configured to control the data processing apparatus to use state switching branch instructions that serve to switch to execution of Java bytecodes starting from a specified memory address location.

62. (Previously Presented) A computer program product as claimed in claim 43, wherein when the code is executed by the data processing apparatus, a program to be executed is stored within fragmented memory regions within said memory.